Southern Technical University
ThiQar Technical College

Electromechanical Systems Techniques Engineering Department

\: Advance Programming

>> 2nd Year

Assist Lect. Ahmed A.J Al-Hchaimi
M.Tech of CNIS
CCNA, MCSA, C, C++, JAVA, VB.NET, VB.6, MATLAB, ORACLE

\: Advance Programming By >> C++

Course Structure

1. Programming in C++ Language :

- An Introduction.
- Programs structure & output.
- Variables and Strings.
- Comments.
- Keywords.
- Variables Declaration.
- Program body.
- Integers in C++.
- Increment and Decrement.

2. Conditions Instructions :

- Inputs.
- (IF. . . Else) statement.
- Complex instructions.
- Complex conditions.
- Boolean's expressions.
- Rating operands.
- (Switch) statement.
- (Scope) operand.

3. Integer Numbers :

- Types of integers.
- Integer numbers conversions.

4. Loops and Real Number Types :

- (While) – Loop.
- (Do . . . While) – Loop.
- (For) – Loop.
- (Break) – command.
- (Continue) – command.
- (Goto) – command.
- Real numbers.

5. Math Functions and Libraries in C++ :

- Functions by user.
- Functions declarations.
- Test programs.
- Privat variables.
- (Void ()) function.
- Boolean functions.
- Input/output functions.
- (Main ()) function.
- (Exit ()) function.

6. Arrays in C++ :

- Introduction.
- Arrays processing.
- 1-D array initialization.
- Arrays indexing.
- Multi dimensions arrays.

7. Pointers and References in C++ :

- Introduction.
- Pointers.
- References.
- Arrays with pointers.
- Pointers for functions.

8. Strings and Files in C++ :

- Introduction to strings.
- Array of strings.
- Introduction to files.
- Commands of files access.

L1-1 History of C++

C++, as the name implies, is essentially based on the C programming language.

Therefore, it seems prudent to begin with a brief history of C. The C programming language was devised in the early 1970s at Bell Laboratories by Dennis Ritchie. It was designed as a system implementation language for the Unix operating system. The history of C and Unix are closely intertwined. For this reason a lot of Unix programming is done with C. To some extent, C was originally based on the type less language BCPL;

L1-2 C++ Fundamentals

C++ is a programming language. As such, it shares a number of similarities with other programming languages. First we may need to clarify what a programming language is. A computer thinks in 1’s and 0’s. Various switches are either on (1’s), or off (0’s). Most humans, however, have trouble thinking in 1’s and 0’s. A programming language is a language that provides a bridge between the computer and human beings. A “low-level” language is one that is closer to a computer’s thinking than to a human language. A prime example would be Assembly language. A “high-level"language is one that is closer to human language.

Figure 1 Programs are written to handle data.

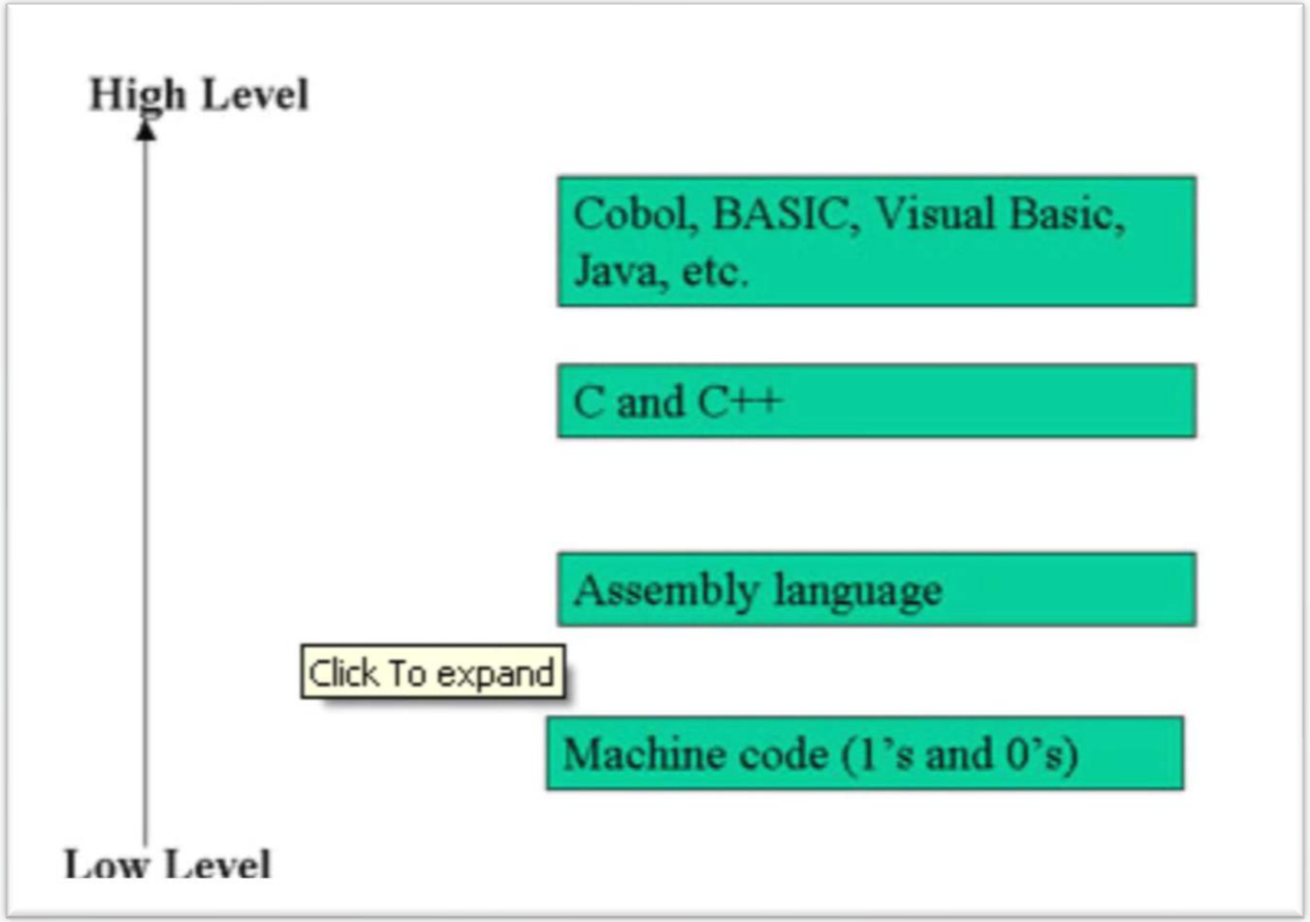

High Level
Cobol, BASIC, Visual Basic, Java, etc.
C and C++
Assembly language
Click To expand
Machine code (1's and 0's)
Low Level

Figure 1: Programming languages.

A variable is simply a place in memory set aside to hold data of a particular type. It is a specific section of the computer's memory that has been reserved to hold some data. It is called a variable because its value or content can vary.

int j;

Then, you have just allocated four bytes of memory; you are using the variable j to refer to those four bytes of memory. You are also stating that the only type of data that j will hold, is whole numbers. (The int is a data type that refers to integers, or whole numbers.) Now, whenever you reference j in your code, you are actually referencing the contents being stored at a specific address in memory.

L1-3 Getting Ready to Program

Programs are written to instruct machines to carry out specific tasks or to solve specific problems. A step-by-step procedure that accomplishes a desired task is called an *algorithm*. Thus, programming is the activity of communicating algorithms to computers.

The Programming Process

1. Specify the task.
2. Discover an algorithm for its solution.
3. Code the algorithm in C++.
4. Test the code

A computer is a digital electronic machine composed of three main components:

Processor, Memory, and Input/Output devices.

The processor is also called the *central processing unit*, or *CPU*. The processor carries out instructions that are stored in the memory. Along with the instructions, data also is stored in memory.

The processor :

typically is instructed to manipulate the data in some desired fashion. Input/output devices take information from agents external to the machine and provide information to those agents. Input devices are typically terminal keyboards, disk drives, and tape drives. Output devices are typically terminal screens, printers, disk drives, and tape drives. The physical makeup of a machine can be quite complicated, but the user need not be concerned with the details. When a simple program is compiled , three separate actions occur:

First

the preprocessor is invoked, then the compiler, and finally the linker. The preprocessor modifies a copy of the source code by including other files and by making other changes. The compiler translates this into *object code*, which the linker then uses to produce the final executable file. A file that contains object code is called an *object file*.

Object files,

unlike source files, usually are not read by humans. When we speak of compiling a program, we really mean invoking the preprocessor, the compiler, and the linker. For a simple program, this is all done with a single command. After the programmer writes a program, it has to be compiled and tested. If modifications are needed, the source code has to be edited again. Thus, part of the programming process consists of this cycle: When the programmer is satisfied with the program performance, the cycle

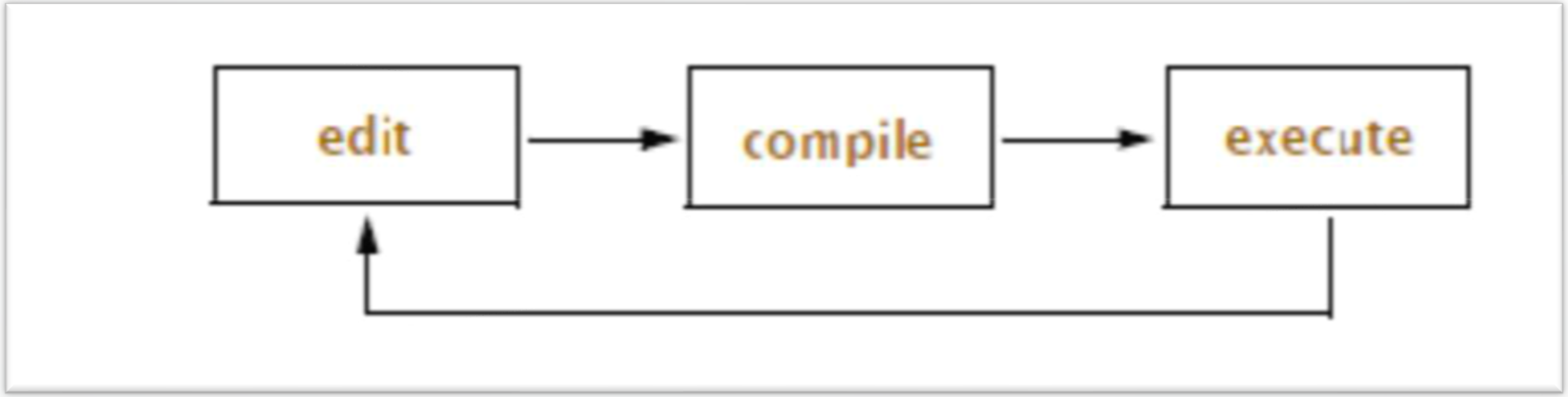


Figure 2: Program Life Cycle.

L1-4 A First Program

A first task for anyone learning to program is to print on the screen. Let us begin by writing the traditional first C++ program which prints the phrase *Hello, world!* On the screen. The complete program is

```
// my first program in C++
#include <iostream>
int main ()
{
cout << "Hello World!";
return 0;
}
```

```
Hello World!
```

The first panel shows the source code for our first program. The second one shows the result of the program once compiled and executed. The previous program is the typical program that programmer apprentices write for the first time, and its result is the printing on screen of the "Hello World!" sentence. It is one of the simplest programs that can be written in C++, but it already contains the fundamental components that every C++ program has.

// my first program in C++

This is a comment line. All lines beginning with two slash signs (//) are considered comments and do not have any effect on the behavior of the program. The programmer can use them to include short explanations or observations within the source code itself. In this case, the line is a brief description of what our program is.

#include <iostream>

Lines beginning with a hash sign (#) are directives for the preprocessor. They are not regular code lines with expressions but indications for the compiler's preprocessor. In this case the directive #include <iostream> tells the preprocessor to include the iostream standard file. This specific file (iostream) includes the declarations of the basic standard input-output library in C++, and it is included because its functionality is going to be used later in the program.

int main ()

This line corresponds to the beginning of the definition of the main function. The main function is the point by where all C++ programs start their execution, independently of its location within the source code.

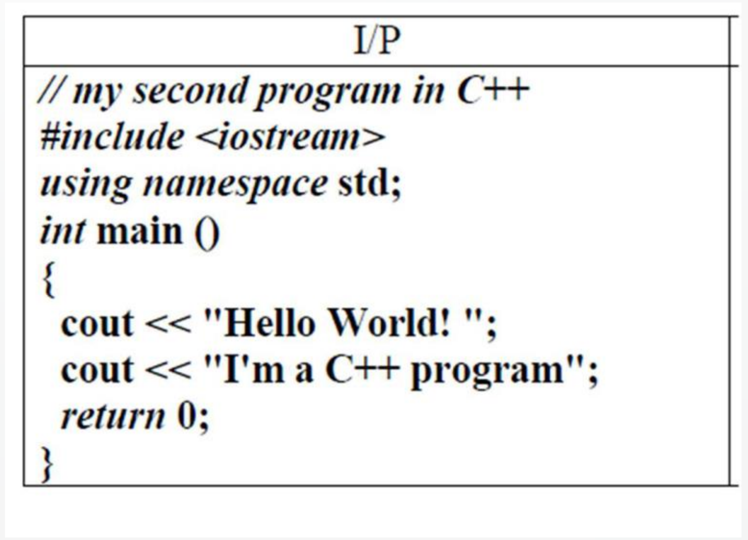
cout << "Hello World";

This line is a C++ statement. A statement is a simple or compound expression that can actually produce some effect. In fact, this statement performs the only action that generates a visible effect in our first program. **cout** represents the standard output stream in C++, and the meaning of the entire statement is to insert a sequence of characters (in this case the Hello World sequence of characters) into the standard output stream (which usually is the screen).

return 0;

The return statement causes the main function to finish. Return may be followed by a return code (in our example is followed by the return code 0). A return code of 0 for the main function is generally interpreted as the program worked Let us add an additional instruction to our first program:

Hello World! I'm a C++ program >>



I/P

```
// my second program in C++
#include <iostream>
using namespace std;
int main ()
{
  cout << "Hello World! ";
  cout << "I'm a C++ program";
  return 0;
}
```